DIGITAL PROCESS OF IMAGES WITH MATLAB

Eduardo Moreno Moreno
Technical engineer telecommunication
## Index

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td>1</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>2</td>
</tr>
<tr>
<td>1.1 Objective</td>
<td>2</td>
</tr>
<tr>
<td>1.2 The utility of this program</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Process</td>
<td>3</td>
</tr>
<tr>
<td>1.4 Matlab 7.0</td>
<td>6</td>
</tr>
<tr>
<td>2. Concepts</td>
<td>8</td>
</tr>
<tr>
<td>2.1 Bwlabel</td>
<td>8</td>
</tr>
<tr>
<td>2.2 Bwselect</td>
<td>9</td>
</tr>
<tr>
<td>2.3 Imerode</td>
<td>10</td>
</tr>
<tr>
<td>2.4 Imdilate</td>
<td>12</td>
</tr>
<tr>
<td>2.5 Imclose</td>
<td>13</td>
</tr>
<tr>
<td>2.6 Imopen</td>
<td>14</td>
</tr>
<tr>
<td>2.7 Regionprops</td>
<td>15</td>
</tr>
<tr>
<td>2.8 Imrotate</td>
<td>16</td>
</tr>
<tr>
<td>3. Procedure</td>
<td>16</td>
</tr>
<tr>
<td>3.1 Part1.m</td>
<td>16</td>
</tr>
<tr>
<td>3.2 Part2.m</td>
<td>17</td>
</tr>
<tr>
<td>3.3 Part3.m</td>
<td>21</td>
</tr>
<tr>
<td>3.4 Calculate limit</td>
<td>23</td>
</tr>
<tr>
<td>3.5 How rotate</td>
<td>25</td>
</tr>
<tr>
<td>3.6 Rotate1 (Rotate2)</td>
<td>26</td>
</tr>
<tr>
<td>3.7 Quit borders</td>
<td>28</td>
</tr>
<tr>
<td>3.8 Margin</td>
<td>30</td>
</tr>
<tr>
<td>3.9 Marco</td>
<td>30</td>
</tr>
<tr>
<td>3.10 Main program</td>
<td>31</td>
</tr>
<tr>
<td>4. Noise</td>
<td>33</td>
</tr>
<tr>
<td>4.1 Using linear filtering</td>
<td>33</td>
</tr>
<tr>
<td>4.2 Using median filtering</td>
<td>34</td>
</tr>
<tr>
<td>4.3 Using adaptive filtering</td>
<td>37</td>
</tr>
<tr>
<td>4.4 Case</td>
<td>38</td>
</tr>
<tr>
<td>4.5 Conclusions</td>
<td>43</td>
</tr>
<tr>
<td>5. Probability and redundancy in the information of the images</td>
<td>44</td>
</tr>
<tr>
<td>5.1 Entropy</td>
<td>44</td>
</tr>
<tr>
<td>5.2 Entropy, redundancy and probability in our images</td>
<td>44</td>
</tr>
<tr>
<td>5.3 Conclusions</td>
<td>48</td>
</tr>
<tr>
<td>6. General conclusions</td>
<td>49</td>
</tr>
<tr>
<td>7. Quote</td>
<td>50</td>
</tr>
</tbody>
</table>
MEMORY
1. Introduction.

1.1 Objective.

The aim of the Project is the digital process of images of the pages of a book scanned improving their quality and manipulating them for their publication in the web or digitally. The software we’re going to use is Matlab 7.0. The techniques we are going to use (with this software) are morphological operations (close, open…), Neighborhood operations, Linear filtering and filter design.

![Figure 1](image1.png)

Figure 1

We can see in the figure 1 the kind of images which we are going to work.

1.2 The utility of this program

The utility of this program for a company is when we want to scan the pages of a book and we want to publish them digitally. In that case we should eliminate the borders of the pages, we should separate the pages, rotate the images if it’s necessary… these techniques are required if we want to represent the pages of the book correctly.

Normally we should process each image individually but if we have for example 2000 images it would be very stressful. With this program we would earn much time, the computer would work more and people would work less.

This program separate the two pages, it rotates them if it’s necessary with the angle which is necessary and it eliminate the borders of the book.
For example we have this scanned image.

The program will process the image and it will show us this result:
1.3 Process.

Next I’m explaining the process for improving the images; but before that it would help us look at the figure of the right. First we’ll find the necessary angle to turn the image (if it’s necessary). When we find the angle we’ll rotate the image and then we’ll eliminate the center of the image separating the images (red part), then we’ll erase the borders of the book (green part).

In the next figure (figure 3) we have a case in which it would be necessary to rotate the image.

Next we’re going to show a scanned image and the final result (left page and right page), next page:
5
1.4 Matlab 7.0.

Matlab (MATrix LABoratory) is mathematical software with an integrated development (IDE) with a language itself. With Matlab we can manipulate matrices, represent data and functions, implement algorithms, it creates interfaces with the user (GUI) and communication with programs of other languages (like C or FORTRAN).

Matlab manipulates the images like matrices, and each color image will be 3 matrices, each one of these matrices will the red component, green component and blue component.

```matlab
A=imread('kairan40.jpg');
A_red=A(:,:,1);
A_green=A(:,:,2);
A_blue=A(:,:,3);

subplot(2,2,1),imshow(A);title('Color Image');
subplot(2,2,2),imshow(A_red);title('Red Component');
subplot(2,2,3),imshow(A_green);title('Green Component');
subplot(2,2,4),imshow(A_blue);title('Blue Component');
```

Applying the last commands we have the next figure (figure 4):
With Matlab we can represent the images like matrixes whose data are integers between 0 and 255 (uint8). Or we can change the type and the images are double (between 0 and 1). White is 255 (or 1) and black is 0.

The Figure 5 shows how Matlab treats the image, besides we have the two kinds (uint8 and double). With uint8 we encode the image with 8 bits (256 levels).

Another kind of images are logical, only ones or zeros. We can see in figure 6 a logical image:
This kind of images only can have two values; 0 or 1, the black areas are 0s and the white areas are 1s.

Besides we can create function with Matlab, for example we can create a function whose name is sum which makes the sum of the elements of a string. I we have the string (1 2 5 3);

We can create the next function:

```matlab
function result = sum(A);
result = 0;
l = length(A);
for i = 1:l
    result = result + A(i);
end
result
```

These functions have to be saved like files with the extension m (M-files).

Now we write in Matlab:

```matlab
string = [1 2 5 3];
res = sum(string);

res =
11
```

2. Concepts.

2.1 Bwlabel.

We can only apply this function to logical images. Bwlabel selects the objects which are not connected in the images. Objects are the areas of ones in the image. For example if we have the logical matrix BW:

```
BW =
1 1 1 0 0 0 0 0
1 1 1 0 1 1 0 0
```
And then we put the next command:

\[ Lab = \text{Bwlabel} \ (BW); \]

The matrix Lab is:

\[
\begin{bmatrix}
1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\
1 & 1 & 1 & 0 & 0 & 1 & 1 & 1
\end{bmatrix}
\]

If we put now the next command: \( \text{max(max(Lab))} \); we find the max of the matrix, or in other side we can find the number of elements as well.

**2.2 Bwselect.**

First we use the Bwlabel function and next we select some points in the image, and then with this function we put:

\[ \text{Bwselect}(BW1,120,58); \]

Selects the object which has the point (120, 58). 120 is the number of column and 58 is the number of row. For example we have the figure 7:
If we put

\( Bwselect(\text{figure}_7, 250, 186) \)

We have then:

In the white area where we have in figure 8 there will be the pixel (186,250); the row 186 and the column 250. Besides we can put vector, in this way we can select many pixels, for example a column or a row (part2.m).

2.3 Imerode.

With this function we ’re erasing parts of white area, so dark area is bigger. Only we can use it with logical functions. Next we show an example:
For example, figure 9 will be named photo. Next we do:

\[ Strel=\text{ones}(10,10); \] (Se is the element which we will use to erode, is a matrix of 1s with 10 rows and 10 columns).

\[ Ero1=\text{imerode(photo,Strel)}; \]

Next we can see the difference between photo and Ero1.

Next, we’ll see different erodes with different sizes in the strels (a bigger strel will create bigger erode).
2.4 Imdilate.

This function makes the contrary of imerode.

\[ Strel=\text{ones}(10,10); \]

\[ Dil=\text{imdilate(photo,Strel)}; \]

Now we difference the original and the dilated image.

Then we’re going to compare with different strels.
2.5 Imclose.

This function makes two operations; first it makes erode and then it makes dilation in the image. Imclose = Imdilate + Imerode.

We have the next picture (photo2):

And we create imclose:

\[ Strel=\text{ones}(10,10); \]
$Closed = \text{imclose}(\text{photo2},\text{Strel});$

With this function the white parts (1s) are closed.

2.6 Imopen.

This function is the contrary of imclose. The white areas are opened.

Besides Imopen = imerode + imdilate.

(Note: In this kind of functions (digital treatment of images) $A+B$ is different of $B+A$).

$Strel = \text{ones}(10,10);$

$Opened = \text{imopen}(\text{photo},\text{Strel});$
We can see that we open white areas, but is different of erode.

2.7 Regionprops.

This function can store many properties of a logical image. First we should bwlabel and then Regionprops. For example:

\[ L = \text{bwlabel}(\text{logical\_image}, 8); \]
\[ \text{STATS} = \text{regionprops}(L, \text{Properties}); \]

PROPERTIES can be a comma-separated list of strings, a cell array containing strings, the string 'all', or the string 'basic'. The set of valid measurement strings includes:


If Properties is the string 'all', then all of the above measurements are computed. If Properties is not specified or if it is the string 'basic', then these measurements are computed: 'Area', 'Centroid' and 'BoundingBox'.

For example:

\[ L = \text{bwlabel}(\text{image}); \]
\[ s = \text{regionprops}(L, \text{'}\text{Area}'\text{}); \]
\[ \text{sup} = \text{cat}(1, s.\text{Area}); \]

Sup is a string which has the areas of each one of the elements of the images. The length of sup is the number of elements of the image. Next we explain the most important properties:

**Area**: Scalar; the actual number of pixels in the region. (This value might differ slightly from the value returned by bwarea, which weights different patterns of pixels differently.)

**BoundingBox** is [ul\_corner width], where ul\_corner is in the form [x y z ...] and specifies the upper left corner of the bounding box and this in the form [x\_width y\_width ...] and specifies the width of the bounding box along each dimension.
**Centroid**: The center of mass of the region. Note that the first element of centroid is the horizontal coordinate (or x-coordinate) of the center of mass, and the second element is the vertical coordinate (or y-coordinate). All other elements of Centroid are in order of dimension.

2.8 Imrotate.

This function rotates an image, for example:

\[ B = \text{Imrotate}(A, \text{Angle}) \] rotates the image \( A \) by \( \text{Angle} \) in degrees.

3. Procedure.

Next I’m going to talk about the procedure which I’ve done. I have created some different functions to structure the program.

3.1 Part1.m

In this M-file we have the original image like input and the next image like output:

![Figure 9](image.png)

Looking at the figure 10 we can see that is a binary image (or logical), the white pixels are 1s and the black pixels are 0s. Now we’re going to see the most important code of part1:

\[ B = \text{rgb2gray}(A); \]
\[ C = \text{im2double}(B); \]
\[ D = \text{im2bw}(C, k); \]
\[ E = \text{not}(D); \]

(NOTE: A is the original color image).

The first function converts a color image to a grayscale image whose values are between 0 and 255. Next function (im2double) changes the interval of the values, I mean B has values between 0 and 255 but C will have values between 0 and 1, so C is a double image. The function im2bw converts a double image (C) to binary image (D). Besides k is the threshold value. I mean if a value is lower than k that value will be 0 in the new image, but if that value is higher than k then that value will be 1 in the new image. The last instruction is very simple; if a pixel in D is 1 in E, it will be 0 in E; and if a pixel in D is 0, then it will be 1 in E. In the next figure we can see every image in this function:

3.2 Part2.m

In this M-file we will eliminate the central area, I mean the separation between pages. The most important function is bwselect. We have a logical image like input and we will create 20 columns and 2 rows for selecting the area of the centre. Next we have a nice example of an input image in this case:
The next figure shows the parts we choose for discriminating the area we don’t need. The green areas are the two rows we choose and the red area represents the 20 columns which are in the centre of the image and every one is separated by 5 spaces.

Next we have the code of the up row:

(NOTE: N is the number of columns of the image, M is the number of rows of the image and fils uses to be 150).

\[
\begin{align*}
\text{for } i &= 1:N \\
\quad r(i) &= \text{fils} \\
\text{end;}
\end{align*}
\]

\[
\begin{align*}
\text{for } i &= 1:N
\end{align*}
\]
\[ c(i) = i; \]

end;

resto0 = bwselect(E, c, r);

Now we show the code of the under row:

for \( i = 1:N \)
    \[ r1(i) = M - \text{fils}; \]
end;

for \( i = 1:N \)
    \[ c1(i) = i; \]
end;

resto1 = bwselect(E, c1, r1);

Next we show the 20 columns.

for \( i = 1:M \)
    \[ g2(i) = i; \]
end;

for \( j = 0:5:100 \)
    for \( i = 1:M \)
        \[ m2((j/5)+1,i) = (\text{round}((N/2)-50)) + j; \]
    end;
end
end

r2 = [g2, g2, g2, g2, g2, g2, g2, g2, g2, g2, g2, g2, g2, g2, g2, g2, g2, g2, g2, g2];
c2 = [m2(1,:), m2(2,:), m2(3,:), m2(4,:), m2(5,:), m2(6,:), m2(7,:), m2(8,:), m2(9,:), m2(10,:), m2(11,:), m2(12,:), m2(13,:), m2(14,:), m2(15,:), m2(16,:), m2(17,:), m2(18,:), m2(19,:), m2(20,:)];

sel1 = bwselect(E, c2, r2);

Now we are going to make an example:
If we make bwselect with the upper row we have as result:

Now if we make bwselect with the down row we have the next figure:
Now we’re going to make bwselect using the 20 columns in the central area, so we have the next result in this case:

![Image of a page divided into two parts]

3.3 Part3.m

In this function we separate the image in two parts; each part is one of the two pages (right part and left part). The first nine lines we have two erodes; we erode the image with a vertical line and then with a horizontal line. This is important because in this way we separate different components.

```matlab
se=strel('line',10,90);
resto1=imerode(resto,se);
se=strel('line',10,0);
resto2=imerode(resto1,se);
imshow(resto2)
```

Next, the lines between 13 and 22 we have a code in which we choose the region whose area is the biggest and then we calculate its boundingbox.

```matlab
L=bwlabel(resto2,8);
STATS=regionprops(L);
areas=(cat(1,STATS.Area))';
```
coordenadas=cat(1,STATS.BoundingBox);
for i=1:length(areas)
    if areas(i)==max(areas)
        indice=i;
    else
        end
end
caja=coordenadas(indice,:);
Finally we use caja(1) and caja(3) (the pixel limit on the right and the pixel limit on the left) due to the help of the boundingbox, and we separate the pages using this limits.

3.4 Calculate limit right (left).m

This function calculates the necessary limit for having a binary image like this:

![Image](image.png)

With this image, which will be called im3, we will able to calculate the necessary angle for rotating the image (if it’s necessary) and the limits for selecting only the page erasing the borders of the book.

For creating this image we have used morphological operations like `imclose`, changes of kinds of images like `im2double`, and we have used the command `im2bw(image, limit)`. The problem of this command is that each image has a different limit for calculating im3 correctly.

For this we have made a boucle in which the limit would start in 0.7 and it will be increasing until we have a correct image like im3. The condition is that it would be a correct limit if we would have some elements in the image because if the image isn’t correctly there will be no elements in the image.

After this boucle we’ll have another, but the condition of this boucle will be that the element we have in the image (im3) has to have an area greater than 200000 pixels.

For example we have the next image:
In this part we obtain the next figure (im3):

![Image](image.png)

The element of this image (white part) will have more than 200000 pixels. Normally those elements in im3 would have around 500000 pixels.
3.5 How rotate right (left).m

With this function we’ll know how to rotate the image. For example we work with im3.

Next step is to know the orientation, if it’s negative (<0) we will turn it in different way
if the orientation is positive (>0). We can obtain the orientation knowing that is a
property of regionprops.

For example next we have two images:

Image A will have a negative orientation and image B will have a positive orientation.

```
L = bwlabel(im3);
s = regionprops(L,'Orientation');
or = cat(1,s.Orientation);
if ori>0
    sense_right=['NOSAR'];
else ori<0
    sense_right=['SAR'];
end
```
In this code we’ll implement an “if”. If orientation is positive (> 0) then we create a variable of characters (sense_right) which value will be NOSAR. However if orientation is negative (< 0) then sense_right will be SAR.

3.6 Rotate1 (or Rotate2).m

This function rotates the image but, what sense does it uses? It depends on the value of sense right (sense left) that may be SAR or NOSAR.

We use the im3 for knowing the angle (right figure).

Next step is to find out the Boundingbox of the elements and in this way we’ll know the coordinates of the next two points (in red).

In this way we can find out the angle using trigonometry in the next red triangle.
In the last two lines of the code we recuperate the original size of the image because when we rotate the image, this changes of size.

Rotate1 will turn the images which are in this way:

And Rotate2 will turn the images which are in this way:
3.7 Quit borders left.m (or right)

When have to find out the Bounding box of the im3 (right figure but now this im3 is turned).

\[ L = bwlabel(im3); \]

\[ s = regionprops(L); \]

\[ Box = cat(1,s.BoundingBox); \]

\[ left = round(Box(1)); \]

\[ up = round(Box(2)); \]

\[ under = round(Box(2)+Box(4)); \]

\[ final_l = image(up:under;left:N::); \]

With the Bounding box we can know the limits of the page and select them, so we select that limits and we’ll have only the page.

First we have next image:
And with Quit borders (left in this case) we’ll have the next result:
3.8 Margin left.m (or right)

In this function we just correct the margins if we have to cut part of the image.

function
[result_left]=margin_left(image,left,up,under);

[M N H]=size(image);

result_left=image(up:under;left:N:);

Values left, up, and under are the number of pixels which cut the three sides of the image.

The figure of the right shows us the red rectangles and the green rectangle which represent the number of pixels we can cut in the image.

3.9 Marco

This function puts a frame in the final result; this means that if we have these final results:
With the function marco we’ll have the next image:

![Image](image.jpg)

### 3.10 Main Program

Now we show the main program with the functions which are explained.

```matlab
A = imread('image.jpg');
image1 = part1(A, 0.75);
image2 = part2(image1, 20);
[res1, res2] = part3(image2, A, 30, 30);
```

Until here we have separated the two pages (left image and right image).

```matlab
[im3_left] = Calculate_limit_left(res1);
[im3_right] = Calculate_limit_right(res2);
```

```matlab
[sense_left] = how_rotate_left(im3_left);
```
[sense_right]=how_rotate_right(im3_right);

In this part we have calculated the image “im3” (im3 left and im3 right) which will help us for the next steps. Besides in how rotate we have found out the sense to rotate the images.

if sense_left==[‘SAR’]
    if sense_right==[‘SAR’]
        turned=rotate_1(A,im3_left,im3_right);
    else sense_right~=[‘SAR’]
        turned=A;
    end
else sense_left==[‘NOSAR’]
    if sense_right==[‘NOSAR’]
        turned=rotate_2(A,im3_left,im3_right);
    else sense_right~=[‘NOSAR’]
        turned=A;
    end
end

In this part, we have made an IF in which if the sense of the images (left and right) is the same we would rotate the original image but unless the sense of the images are the same we wouldn´t rotate the original.

imageA=part1(turned,0.75);
imageB=part2(imageA,20);
close all
[resA,resB]=part3(imageB,turned,C1/2,20);

[im3_l]=Calculate_limit_left(resA);
\[im3\_r]=\text{Calculate\_limit\_right}(\text{resB});

\text{image\_left}=\text{quit\_borders\_left}(\text{im3\_l});

\text{image\_right}=\text{quit\_borders\_right}(\text{im3\_r});

\text{result\_left}=\text{margin\_left}(\text{image\_left},25,30,25);

\text{result\_right}=\text{margin\_right}(\text{image\_right},25,30,25);

The last part of the program consists in working with the rotated image (turned).

4. **Noise**

Digital images are prone to a variety of types of noise. There are several ways that noise can be introduced into an image, depending on how the image is created. For example:

If the image is scanned from a photograph made on film, the film grain is a source of noise. Noise can also be the result of damage to the film, or be introduced by the scanner itself.

If the image is acquired directly in a digital format, the mechanism for gathering the data (such as a CCD detector) can introduce noise.

Electronic transmission of image data can introduce noise.

Matlab provides a number of different ways to remove or reduce noise in an image. Different methods are better for different kinds of noise. The methods available include:

1) Using Linear Filtering
2) Using Median Filtering
3) Using Adaptive Filtering

4.1 **Using Linear Filtering**

You can use linear filtering to remove certain types of noise. Certain filters, such as averaging or Gaussian filters, are appropriate for this purpose. For example, an averaging filter is useful for removing grain noise from a photograph. Because each pixel gets set to the average of the pixels in its neighborhood, local variations caused by grain are reduced.
4.2 Using Median Filtering

Median filtering is similar to using an averaging filter, in that each output pixel is set to an average of the pixel values in the neighborhood of the corresponding input pixel. However, with median filtering, the value of an output pixel is determined by the median of the neighborhood pixels, rather than the mean. The median is much less sensitive than the mean to extreme values (called outliers). Median filtering is therefore better able to remove these outliers without reducing the sharpness of the image. The medfilt2 function implements median filtering.

Note: Median filtering is a specific case of order-statistic filtering, also known as rank filtering.

We show the following example:

```matlab
Imagen = imread('kairan37.jpg');
I=rgb2gray(Imagen);
subplot(1,2,1),imshow(Image);title('IMAGE (COLOR)');
subplot(1,2,2),imshow(I);title('I (GRAYSCALE)');
```

```matlab
J = imnoise(I,'salt & pepper',0.02);
figure, imshow(J);title('IMAGE WITH NOISE');
```
We use averaging filter (K).

\[ K = \text{filter2}('\text{average}',3,J)/255; \]

We use now median filter.

\[ L = \text{medfilt2}(J,[3 3]); \]

\[ \text{subplot}(1,2,1), \text{imshow}(K); \text{title('averagin')}; \]
\[ \text{subplot}(1,2,2), \text{imshow}(L); \text{title('median')}; \]
We can see that this filters (averaging and median) work of the next way:

fspecial('average',3) is a matrix which size is 3x3 and calculates the mean of the pixels of the image, this matrix would be:

\[
\begin{bmatrix}
0.1111 & 0.1111 & 0.1111 \\
0.1111 & 0.1111 & 0.1111 \\
0.1111 & 0.1111 & 0.1111 \\
\end{bmatrix}
\]

If we add every component we obtain 1.

The command \[ K = \text{filter2}(\text{fspecial('average',3)}, J)/255; \] creates the previous matrix and it acts like a mask. For example if we have the next matrix:

\[
\begin{bmatrix}
2 & 3 & 1 & 3 \\
2 & 5 & 4 & 1 \\
3 & 9 & 6 & 4 \\
7 & 2 & 6 & 0 \\
\end{bmatrix}
\]

The result of filtering this matrix with the mask of mean is:

\[
\begin{bmatrix}
1.3333 & 1.8889 & 1.8889 & 1.0000 \\
2.6667 & 3.8889 & 4.0000 & 2.1111 \\
3.1111 & 4.8889 & 4.1111 & 2.3333 \\
2.3333 & 3.6667 & 3.0000 & 1.7778 \\
\end{bmatrix}
\]

For example: \[ G(2,3) = m(1,2)M(1,2) + m(1,3)M(1,3) + m(1,4)M(1,4) + m(2,2)M(2,2) + m(2,3)M(2,3) + m(2,4)M(2,4) + m(3,2)M(3,2) + m(3,3)M(3,3) + m(3,4)M(3,4). \]
If we see the figure 1 again we can see that the median filter is much more efficient than mean filter (or averaging filter). Next we’re going to explain why it happens.

For example if we have the next matrix:

\[
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 5 & 0 \\
0 & 0 & 0 \\
\end{array}
\]

Number 5 represents the point of noise, if we make the mean filter (3x3) in this matrix we have:

\[
\begin{array}{ccc}
5/9 & 5/9 & 5/9 \\
5/9 & 5/9 & 5/9 \\
5/9 & 5/9 & 5/9 \\
\end{array}
\]

Now if we make the median filter (3x3) we obtain:

\[
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
\end{array}
\]

We have eliminated the number 5 (the noise). With the mean filter we would eliminate the high frequencies.

4.3 Using Adaptive Filtering

The wiener2 function applies a Wiener filter (a type of linear filter) to an image adaptively, tailoring itself to the local image variance. Where the variance is large, wiener2 performs little smoothing. Where the variance is small, wiener2 performs more
smoothing. This approach often produces better results than linear filtering. The adaptive filter is more selective than a comparable linear filter, preserving edges and other high-frequency parts of an image. In addition, there are no design tasks; the wiener2 function handles all preliminary computations and implements the filter for an input image. wiener2, however, does require more computation time than linear filtering. Wiener2 works best when the noise is constant-power ("white") additive noise, such as Gaussian noise. The example below applies wiener2 to an image that has had Gaussian noise added.

\[
\begin{align*}
RGB &= \text{imread('kairan37.jpg');} \\
I &= \text{rgb2gray(RGB);} \\
J &= \text{imnoise(I,'gaussian',0,0.005);} \\
K &= \text{wiener2(J,[5 5]);}
\end{align*}
\]

```matlab
subplot(1,2,1),imshow(J);title('IMAGE WITH NOISE');
subplot(1,2,2),imshow(K);title('IMAGE FILTERED WITH WIENER');
```

4.4 Case

Now we´re going to use some of the techniques with a case of our project; for example this figure (result_right):
If we want to use these techniques of elimination of noise, first of all we have to separate the tree components of the image.

\[
J_{\text{red}} = \text{result}_{\text{right}}(:,:,1);
\]

\[
J_{\text{green}} = \text{result}_{\text{right}}(:,:,2);
\]

\[
J_{\text{blue}} = \text{result}_{\text{right}}(:,:,3);
\]

First we’ll use the averaging filter (mean):

\[
K1 = \text{filter2(fspecial('average',3)},J_{\text{red}})/255;
\]

\[
K2 = \text{filter2(fspecial('average',3)},J_{\text{green}})/255;
\]

\[
K3 = \text{filter2(fspecial('average',3)},J_{\text{blue}})/255;
\]

We join the tree components:

\[
\text{image} = \text{cat}(3,K1,K2,K3);
\]

\[
\text{imshow(image)}
\]

This is the result:
We can see that this process doesn´t change the image.

Now we’re going to try with a median filter:

$L1 = medfilt2(J\_red,[3\ 3]);$

$L2 = medfilt2(J\_green,[3\ 3]);$

$L3 = medfilt2(J\_blue,[3\ 3]);$

$\text{image2=cat}(3,L1,L2,L3);$  

$\text{imshow(image2)}$
In this case the image doesn’t change either.

We ’ll try it with the third option: wiener filter.

\[
W1 = \text{wiener2}(J\_red,[5\ 5]);
\]

\[
W2 = \text{wiener2}(J\_green,[5\ 5]);
\]

\[
W3 = \text{wiener2}(J\_blue,[5\ 5]);
\]

\[
\text{image3} = \text{cat}(3,W1,W2,W3);
\]

\[
\text{imshow(image3)}
\]

We see again that the image doesn’t changes.

We can say that these points we appreciate in the image are not noise, it looks like they are stains of the paper.

But now, if we try this kind of filter:

\[
K1 = \text{filter2(fspecial('average',10), J\_red)/255;}
\]

The filter has 10x10 sizes. We have this:
We can see that if we use a big filter we’ll lose information in the image.

Now we’re going to try with a big median filter (10x10):

We see again that we’ll lose information if we use a big median filter.
Finally, if we use a big wiener filer we’ll obtain this:

We’ll lose information again

4.5 Conclusions

We can see that depending on the kind of noise we have, we should use different kind of techniques to eliminate the noise. Each technique is created for each kind of noise. We have to know that there are many kinds of noises; for example we have the noise which is created by the electronic components, or the noise created by the temperature…
5. Probability and redundancy in the information of the images.

5.1 Entropy

The entropy is the quantity of information in a message. Besides the entropy says us how I can compress the dates of that message.

For example if we are waiting for a message and the options are “a b c d e f g h i j k l m n o p q r s t u w v x y z” (26 options) but the message we received is: "qalmnbphijcdgketrsfuvxyzwo", we can see that we don’t have any repeated letter and the order is very random so the entropy will be very low and the compression won’t be efficient. The formula is:

\[ H(X) = E(I(X)) = \sum_{i=1}^{n} p(x_i) \log_2 \left( \frac{1}{p(x_i)} \right) \]

\[ = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i) \]

The entropy is the mathematical limit of the compression which you can do without losing information. We have to say that entropy is in bits.

For example we have a random variable (X) that can have four different values with probabilities (1/2, 1/4, 1/8, 1/8). Now we’re going to use the formula:

\[ H(X) = 1/2*\log_2(2) + 1/4*\log_2(4) + 1/8\log_2(8) + 1/8\log_2(8) = 1.75 \]

This means that we can encode or compress this random variable X with 1.75 bits, but we have to use integer values, so we should use 2 bits. Besides this means that the random variable X can be compressed with 2 bits without losing information, but if we compress it with 1 bit we’ll lose information, so the limit will be 2 bits (exactly 1.75 bits).

5.2 Entropy, Redundancy and probability in our images

Now we’re going to calculate the entropy of the results of the components of color of our images, for example like this:
We have processed 20 images (20 left and 20 right) so we have 40 images. First we have calculated the entropy of the 20 left images and we have created a graph.
We have three graphs; the red graph represents the entropy of the 20 red components of our images. The green graph will be the green components and the blue graph will be the blue components.

We have to comment that the blue component is the component with more entropy of the three. So the blue component should be compressed with more bits than the red and green components.

Now we’re going to make a table like this:

<table>
<thead>
<tr>
<th>Components</th>
<th>Maximum Value</th>
<th>Minimum Value</th>
<th>Bits options</th>
<th>Number of bits recommended in compression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>5.28</td>
<td>4.45</td>
<td>6 and 5</td>
<td>6</td>
</tr>
<tr>
<td>Green</td>
<td>5.19</td>
<td>4.48</td>
<td>6 and 5</td>
<td>6</td>
</tr>
<tr>
<td>Blue</td>
<td>5.54</td>
<td>4.96</td>
<td>6 and 5</td>
<td>6</td>
</tr>
</tbody>
</table>

Normally the images are codified with 8 bits, so we can compress them with 6 bits without losing information. In some cases we could compress the images with 5 bits (in the minimum values for example) without losing information.
Next we show the entropy of the right images (with the three components: Red, green and blue) and we can see that the blue component has more entropy than the other.

Next we’re going to calculate the histograms (3 histograms, one for each component) of one image, for example this:

Now we calculate the histograms:
We are analyzing the graph: The green and red components look like (only the green component is a bit darker than red component), however the blue component is different, first of all we can see that the blue graph is smaller, it means that there isn’t a predominant value like the other components. Besides the blue histogram is wider than the other histograms (which are sharpest). So this means that the blue component is a more random variable. When a variable is more random its entropy will be greater.

5.3 Conclusions

This part has been written because we think that could be important the transmission of the images like signals, we can see that if we use the codification by probability the red and green components could be more compressed than the blue component so the bandwidth of the blue component would be bigger.

However we have to say that these differences are not significant.
6. General conclusions

We have to say that the most difficult part of this project has been the variety of the images; this means that we can separate these images in many groups. Each group has their characteristics, for example the intensity, the size of the image, the quality which is scanned the image… Due to this variety I had to do many boucles, IF … to can process all the possibilities.

The most important question of this project was to try that people work less as possible as the computer works more. For example, in the function Calculate limit and the beginning we you had to choose one limit (0.7 for example) and if the result was not satisfactory (im3) you had to change that limit, but now the program look for that limit, so humans needn´t to select a limit.

In spite of trying that humans work less and machines work more I didn´t want that the efficiency of the results was worst. Between the efficiency of the results and the independence of the machine with the humans if preferred that the efficiency of the results was more important.

About the results I would like to say that the quality of the image is not changed, this project treats to select some areas of the images, separate them and rotate them if it’s possible. These areas are the text of the pages.

It´s necessary to comment that matlab threats the images with less quality but it´s not a problem because at the end we can put a command like this:

\textit{Imwrite (image, ‘path’, ‘quality’, x)};

We keep the image in the directory ‘path’ with a fixed ‘quality’, this is very common in jpeg images.
Quote
<table>
<thead>
<tr>
<th>Item</th>
<th>Price 1 unit</th>
<th>Number of units</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matlab 7.0 Student version</td>
<td>89 $</td>
<td>1</td>
<td>89 $</td>
</tr>
<tr>
<td>Computer Hacer Aspire 5730Z</td>
<td>420 $</td>
<td>1</td>
<td>420 $</td>
</tr>
<tr>
<td>Salary Engineer Tec. Telecommunications</td>
<td>54.4 $/hour</td>
<td>60 hours</td>
<td>3264 $</td>
</tr>
<tr>
<td>Office material (pens, erasers, sheets of paper…)</td>
<td>-</td>
<td>-</td>
<td>20 $</td>
</tr>
</tbody>
</table>

Total: 3793 dollars.
function [im3]=Calculate_limit_left(image)
    AB=image(:,:,3);
    ABD=im2double(AB);
    log=im2bw(ABD,0.7);
    imshow(log)
    [M N]=size(log);
    for i=1:N
        r(i)=i;
    end;
    for i=1:N
        c(i)=30;
    end;
    only=bwselect(log,c,r);
    imshow(only)
    se=strel('disk',15);
    im1=imclose(only,se);
    i=16;
    while im1(M-8:M,1)==0
        se=strel('disk',i);
        im1=imclose(only,se);
        i=i+2;
    end
    imshow(im1)
    im2=not(im1);
    imshow(im2)
    for i=1:50
        r1(i)=(round(M/2));
    end;
    for i=1:50
        c1(i)=N-i;
    end;
    im3=bwselect(im2,c1,r1);
    im3num=bwlabel(im3);
    s=0.71;
    while max(max(im3num))==0
        log=im2bw(ABD,s);
        imshow(log)
        [M N]=size(log);
        for i=1:N
            r(i)=i;
        end;
    end;
for i=1:N  
c(i)=30;  
end

only=bwselect(log,c,r);  
imshow(only)

se=strel('disk',15);  
im1=imclose(only,se);  
i=16;  
while im1(M-8:M,1)==0  
    se=strel('disk',i);  
    im1=imclose(only,se);  
    i=i+2;
end

imshow(im1)  

im2=not(im1);  
imshow(im2)

for i=1:50  
    rl(i)=(round(M/2));  
end;

for i=1:50  
    cl(i)=N-i;  
end;

im3=bwselect(im2,cl,rl);  
im3num=bwlabel(im3);  
s=s+0.01;
end

L = bwlabel(im3);  
struct = regionprops(L);  
areas=cat(1,struct.Area);

while areas(1)<200000  
    log=im2bw(ABD,s);  
    imshow(log)  
    [M N]=size(log);
    for i=1:N  
        r(i)=i;  
    end;
    for i=1:N  
        c(i)=30;  
    end
    only=bwselect(log,c,r);  
    imshow(only)
se=strel('disk',15);
im1=imclose(only,se);
i=16;
while im1(M-8:M,1)==0
    se=strel('disk',i);
im1=imclose(only,se);
i=i+2;
end

imshow(im1)
im2=not(im1);
imshow(im2)

for i=1:50
    r1(i)=(round(M/2));
end;

for i=1:50
    c1(i)=N-i;
end;

im3=bwselect(im2,c1,r1);
im3num=bwlabel(im3);

L = bwlabel(im3);
struct = regionprops(L);
areas=cat(1,struct.Area);
s=s+0.01;
end
function [im3]=Calculate_limit_right(image)

AB=image(:,:,3);

ABD=im2double(AB);
log=im2bw(ABD,0.7);
imshow(log)
[M N]=size(log);

for i=1:N
    r(i)=i;
end;

for i=1:N
    c(i)=N-30;
end;

only=bwselect(log,c,r);
imshow(only)
se=strel('disk',15);
im1=imclose(only,se);
i=16;
while im1(M-8:M,1)==0
    se=strel('disk',i);
im1=imclose(only,se);
i=i+2;
end
imshow(im1)
im2=not(im1);
imshow(im2)

for i=1:50
    r1(i)=(round(M/2));
end;

for i=1:50
    cl(i)=i;
end;

im3=bwselect(im2,cl,r1);
im3num=bwlabel(im3);

s=0.71;
while max(max(im3num))==0
    log=im2bw(ABD,s);
imshow(log)
[M N]=size(log);

    for i=1:N
        r(i)=i;
    end;

    for i=1:N
        c(i)=N-30;
end;
end

only=bwselect(log,c,r);
imshow(only)

se=strel('disk',15);
iml=imclose(only,se);
i=16;
while iml(M-8:M,1)==0
    se=strel('disk',i);
iml=imclose(only,se);
i=i+2;
end

imshow(iml)
im2=not(iml);
imshow(im2)

for i=1:50
    r1(i)=(round(M/2));
end;

for i=1:50
    c1(i)=i;
end;
im3=bwselect(im2,c1,r1);
im3num=bwlabel(im3);
s=s+0.01;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
L = bwlabel(im3);
struct = regionprops(L);
areas=cat(1,struct.Area);

while areas(1)<200000
    log=im2bw(ABD,s);
imshow(log)
[M N]=size(log);
    for i=1:N
        r(i)=i;
end;
    for i=1:N
        c(i)=30;
end
only=bwselect(log,c,r);
imshow(only)
se = strel('disk', 15);
im1 = imclose(only, se);
i = 16;
while im1(M-8:M, 1) == 0
    se = strel('disk', i);
im1 = imclose(only, se);
i = i + 2;
end

imshow(im1)
im2 = not(im1);
imshow(im2)

for i = 1:50
    r1(i) = (round(M/2));
end;

for i = 1:50
    c1(i) = N - i;
end;

im3 = bwselect(im2, c1, r1);
im3num = bwlabel(im3);

L = bwlabel(im3);
struct = regionprops(L);
areas = cat(1, struct.Area);
s = s + 0.01;

end
function [sense_right]=how_rotate_right(im3)

L = bwlabel(im3);
s  = regionprops(L,'Orientation');
ori=cat(1,s.Orientation);
if ori>0
    sense_right=['NOSAR'];
else ori<0
    sense_right=['SAR'];
end
% Main Program

close all
clear all

A=imread('kairan61.jpg'); % 263

% Next i´m going to separate the pages of the image.

image1=part1(A,0.75);
image2=part2(image1,20);
close all
[res1,res2]=part3(image2,A,30,30);

% The function imshow shows us the image.

close all
figure; imshow(res1)
figure; imshow(res2)

% The next part calculates im3 (left and right) which is
% an important image because with it we will able to
% calculate the necessary angle to rotate or to erase the
% borders of the book.

[im3_left]=Calculate_limit_left(res1);
[im3_right]=Calculate_limit_right(res2);

% Next part finds out the sense we have to turn the
% image (if it´s necessary to turn it).

[sense_left]=how_rotate_left(im3_left);
[sense_right]=how_rotate_right(im3_right);

% The next If says that depending on the sense we´ll
% turn the image in different way.

if sense_left=='SAR'
    if sense_right=='SAR'
        turned=rotate_1(A,im3_left,im3_right);
    else sense_right=='NOSAR'
        turned=A;
    end
else sense_left=='NOSAR'
    if sense_right=='NOSAR'
        turned=rotate_2(A,im3_left,im3_right);
    else sense_right=='SAR'
        turned=A;
    end
end

figure; imshow(A)
figure; imshow(turned)

% This part separates the two pages again but this time with
% the turned imaged.
imageA = part1(turned, 0.75);
imageB = part2(imageA, 20);
close all
[resA, resB] = part3(imageB, turned, 20, 20);
close all
figure; imshow(resA)
figure; imshow(resB)

% This part calculates im3 (left and right) again but this
% time with the turned image.

[im3_l] = Calculate_limit_left(resA);
[im3_r] = Calculate_limit_right(resB);

% Quit_borders erases the borders of the book
image_left = quit_borders_left(im3_l, resA);
image_right = quit_borders_right(im3_r, resB);
close all
figure; imshow(image_left)
figure; imshow(image_right)

result_left = margin_left(image_left, 25, 30, 25);
result_right = margin_right(image_right, 25, 30, 25);
close all
figure; imshow(result_left)
figure; imshow(result_right)
figure; imshow(A)

final = marco(result_left, result_right);
imshow(final)
function im2=marco(result_left,result_right)

[M N H]=size(result_left);
[M1 N1 H1]=size(result_right);

if M==M1
else M~=M1
if M<M1
    for i=M+1:M1
        result_left(i,:,:)=127;
    end
else M>M1
    for i=M1+1:M
        result_right(i,:,:)=127;
    end
end
end

[F C H]=size(result_left);

Ver=zeros(F,200);
for i=1:F
    for j=1:200
        Ver(i,j)=127;
    end
end

V=cat(3,Ver,Ver,Ver);
im1=[V,result_left,V,result_right,V];

[F1 C1 H1]=size(im1);
Hor=zeros(100,C1);
for i=1:100
    for j=1:C1
        Hor(i,j)=127;
    end
end

H=cat(3,Hor,Hor,Hor);
im2=cat(1,H,im1,H);
function [result_left]=margin_left(image,left,up,under);

[M N H]=size(image);
result_left=image(up:M-under,left:N,:);
function [result_right]=margin_right(image,right,up,under);

[M N H]=size(image);
result_right=image(up:M-under,1:N-right,:);
function [E]=part1(A,k)

B=rgb2gray(A);
figure(2)
imshow(B)
C=im2double(B); % La paso a binaria (cero y unos)
figure(3)
imshow(C)
%% k=graythresh(C);
D=im2bw(C,k);
figure(4)
imshow(D)
E=not(D);
figure(5)
imshow(E)
function [resto]=part2(E,fils)

% fils uses to be 150.
[M N altura]=size(E);

% First we prepare the up row.
for i=1:N
    r(i)=fils;
end;

for i=1:N
    c(i)=i;
end;

% Now we prepare the under row.
for i=1:N
    r1(i)=M-fils;
end;

for i=1:N
    c1(i)=i;
end;

% Next we prepare the 20 columns.
for i=1:M
    g2(i)=i;
end;

for j=0:5:100
    for i=1:M
        m2((j/5)+1,i)=(round((N/2)-50))+j;
    end;
end

r2=[g2,g2,g2,g2,g2,g2,g2,g2,g2,g2,g2,g2,g2,g2,g2,g2,g2,g2,g2,g2,g2];
c2=[m2(1,:),m2(2,:),m2(3,:),m2(4,:),m2(5,:),m2(6,:),m2(7,:),m2(8,:),m2(9,:),m2(10,:),m2(11,:),m2(12,:),m2(13,:),m2(14,:),m2(15,:),m2(16,:),m2(17,:),m2(18,:),m2(19,:),m2(20,:)];

resto0=bwselect(E,c,r);
resto1=bwselect(E,c1,r1);
sel1=bwselect(E,c2,r2);
sel2=or(resto0,resto1);
resto=and(sel1,sel2);
imshow(resto)
function [half1,half2]=part3(resto,A,marg_iz,marg_der)

[M N altura]=size(A);
se=strel('line',10,90);
resto1=imerode(resto,se);

se=strel('line',10,0);
resto2=imerode(resto1,se);
imshow(resto2)
L=bwlabel(resto2,8);
STATS=regionprops(L);

areas=(cat(1,STATS.Area))';
coordenadas=cat(1,STATS.BoundingBox);

for i=1:length(areas)
    if areas(i)==max(areas)
        indice=i;
    else
        end
end
caja=coordenadas(indice,:);

i=1;
while caja(3)>220
    se=strel('disk',i);
    resto2=imerode(resto1,se);
    imshow(resto2)
    L=bwlabel(resto2,8);
    STATS=regionprops(L);

    areas=(cat(1,STATS.Area))';
    coordenadas=cat(1,STATS.BoundingBox);

    for i=1:length(areas)
        if areas(i)==max(areas)
            indice=i;
        else
            end
    end
    caja=coordenadas(indice,:);
    i=i+1;
end

izq=round(caja(1))-marg_iz;
der=round(caja(1)+caja(3))+marg_der;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
trozo1=A(:,1:izq,:);
trozo2=A(:,der:N,:);
half1=trozo1;
half2=trozo2;
function [final_l]=quit_borders_left(im3,image);

[M N]=size(im3);
im4=im3(:,1:N-40);
L = bwlable(im4);
s = regionprops(L);
Box=cat(1,s.BoundingBox);
left=round(Box(1));
up=round(Box(2));
under=round(Box(2)+Box(4));

final_l=image(up:under,left:N,:);
function [final_r]=quit_borders_right(im3,image);

[M N]=size(im3);
im4=im3(:,40:N);
L = bwlabel(im4);
s = regionprops(L);
Box=cat(1,s.BoundingBox);
right=round(1+Box(3));
up=round(Box(2));
under=round(Box(2)+Box(4));

final_r=image(up:under,1:right,:);
function [rot_left, ang]=rotate_1(A, im3_left, im3_right)

%% Aquí roto en la izquierda, el que sube

[M N]=size(im3_left);
[F C]=size(im3_right);
im4_left=im3_left(:,1:N-40);
[M1 N1]=size(im4_left);

im4_right=im3_right(:,40:C);
[F1 C1]=size(im4_right);

L = bwlabel(im4_left);
s = regionprops(L);
box=cat(1,s.BoundingBox);

C1=round(box(3));
Xa=round(box(1));
C2=0;
i=round(box(2));
while im4_left(i,Xa)==0
    C2=C2+1;
i=i+1;
end
if C1==0
    C1=1;
end

rad1=atan(C2/C1);
ang1=(rad1*360)/(2*pi);

L1 = bwlabel(im4_right);
s1 = regionprops(L1);
box1=cat(1,s1.BoundingBox);

Xb=round(box1(2));
Cat1=0;
i1=Xb;
while im4_right(i1,1)==0
    Cat1=Cat1+1;
i1=i1+1;
end
j1=1;
Cat2=0;
while im4_right(Xb,j1)==0
    Cat2=Cat2+1;
j1=j1+1;
end
if Cat2==0
    Cat2=1;
end

rad2=atan(Cat1/Cat2);
ang2=(rad2*360)/(2*pi);

end
grades(1)=abs(ang1);
grades(2)=abs(ang2);

ang=min(grades);

rotado=imrotate(A,-ang);
%imshow(rotado)

[M2 N2 alt]=size(rotado);
[row col hig]=size(A);

L1=round(abs(M2-row));
L2=round(abs(N2-col));

if L1==0
    L1=1
end
if L2==0
    L2=1
end

output=rotado(L1:M2-L1,L2:N2-L2,:);
rot_left=output;
function rot_left=rotate_2(A,im3_left,im3_right)

%% Aqui roto en la izquierda, el que baja

[M N]=size(im3_left);
[F C]=size(im3_right)
im4_left=im3_left(:,1:N-40);
[M1 N1]=size(im4_left);

im4_right=im3_right(:,40:C);
[F1 C1]=size(im4_right);
L = bwlabel(im4_left);
s = regionprops(L);
box=cat(1,s.BoundingBox);
C1=round(box(3));
C2=0;
i=round(box(2));

while im4_left(i,N1)==0
    C2=C2+1;
i=i+1;
end
if C1==0
    C1=1;
end
rad1=atan(C2/C1);
ang1=(rad1*360)/(2*pi);

L1 = bwlabel(im4_right);
s1 = regionprops(L1);
box1=cat(1,s1.BoundingBox);
Cat2=round(box1(3));
Cat1=0;
i1=round(box1(2));
while im4_right(i1,Cat2)==0
    Cat1=Cat1+1;
i1=i1+1;
end
if Cat2==0
    Cat2=1;
end
rad2=atan(Cat1/Cat2);
ang2=(rad2*360)/(2*pi);
grades(1)=ang1;
grades(2)=ang2;
ang=min(grades);
rotado = imrotate(A, ang);
imshow(rotado)

[M2 N2 alt] = size(rotado);
[row col hig] = size(A);

L1 = round(abs(M2 - row));
L2 = round(abs(N2 - col));

if L1 == 0
    L1 = 1
end
if L2 == 0
    L2 = 1
end

output = rotado(L1:M2-L1, L2:N2-L2,:);
rot_left = output;
function [sense_left]=how_rotate_left(im3)

L = bwlabel(im3);
s  = regionprops(L,'Orientation');
ori=cat(1,s.Orientation);

if ori>0
    sense_left=['NOSAR'];
else ori<0
    sense_left=['SAR'];
end
DIGITAL PROCESS OF IMAGES WITH MATLAB 7.0
AIM OF THE PROJECT

- Manipulating their digital presentation (digital book or website).
- Improving their quality with techniques to erase noise.
- It’s better keep information in computers than in papers.
KIND OF IMAGES
KIND OF IMAGES
KIND OF IMAGES
UTILITY OF THIS PROGRAM

- When a company wants to publish them digitally (website for example).
- Humans would work less and computers would work more.
- We would earn much time with this program.
MATLAB 7.0
MATLAB 7.0

- Mathematical software with a language itself.
- It can implement algorithms, functions and it can communicate with other languages (C or FORTRAN).
- Images like matrices.
# MATLAB 7.0

## Uint8

- Between 0 and 255

<p>| | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>127</td>
<td>127</td>
<td>127</td>
<td>127</td>
<td>127</td>
<td>127</td>
<td>127</td>
<td>127</td>
<td>127</td>
<td>127</td>
<td>127</td>
</tr>
<tr>
<td>127</td>
<td>127</td>
<td>127</td>
<td>127</td>
<td>127</td>
<td>127</td>
<td>127</td>
<td>127</td>
<td>127</td>
<td>127</td>
<td>127</td>
</tr>
<tr>
<td>127</td>
<td>127</td>
<td>127</td>
<td>127</td>
<td>127</td>
<td>127</td>
<td>127</td>
<td>127</td>
<td>127</td>
<td>127</td>
<td>127</td>
</tr>
<tr>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
</tr>
</tbody>
</table>
Double

- Between 0 and 1.

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
</tr>
<tr>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
</tr>
<tr>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
</tr>
<tr>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
</tbody>
</table>
### MATLAB 7.0

<p>| | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
</tr>
<tr>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
</tr>
<tr>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.5000</td>
</tr>
<tr>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
</tbody>
</table>
Logical or binary

- Only 0 or 1
MATLAB 7.0

RGB images

- Color images

- 3 dimensions:
  - Red component (matrix Red)
  - Green component (matrix Green)
  - Blue component (matrix Blue)
<table>
<thead>
<tr>
<th></th>
<th>Blue</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.2235</td>
<td>0.2902</td>
<td><strong>0.0627</strong></td>
<td>0.2902</td>
<td>0.2902</td>
</tr>
<tr>
<td></td>
<td>0.5804</td>
<td>0.0627</td>
<td>0.0627</td>
<td>0.2235</td>
<td>0.2588</td>
</tr>
<tr>
<td></td>
<td>0.5176</td>
<td>0.1922</td>
<td>0.0627</td>
<td><strong>0.1608</strong></td>
<td>0.1922</td>
</tr>
<tr>
<td></td>
<td>0.5176</td>
<td>0.1294</td>
<td><strong>0.1608</strong></td>
<td>0.1294</td>
<td>0.2588</td>
</tr>
<tr>
<td></td>
<td>0.5176</td>
<td>0.1608</td>
<td>0.0627</td>
<td>0.1608</td>
<td>0.1922</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Green</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.1922</td>
<td>0.2588</td>
<td>0.2588</td>
<td>0.2588</td>
<td>0.2588</td>
</tr>
<tr>
<td></td>
<td>0.1294</td>
<td><strong>0.1608</strong></td>
<td>0.1294</td>
<td>0.2588</td>
<td>0.2588</td>
</tr>
<tr>
<td></td>
<td>0.1294</td>
<td>0.1608</td>
<td>0.1294</td>
<td>0.2588</td>
<td>0.2588</td>
</tr>
<tr>
<td></td>
<td>0.1608</td>
<td>0.1608</td>
<td>0.1922</td>
<td>0.2588</td>
<td>0.2588</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Red</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.7412</td>
<td>0.7765</td>
<td>0.7765</td>
<td>0.7765</td>
<td>0.902</td>
</tr>
<tr>
<td>490</td>
<td>0.3882</td>
<td><strong>0.5176</strong></td>
<td>0.5804</td>
<td>0.5804</td>
<td>0.7765</td>
</tr>
<tr>
<td></td>
<td>0.2588</td>
<td>0.2588</td>
<td>0.2235</td>
<td>0.4824</td>
<td>0.2235</td>
</tr>
<tr>
<td></td>
<td>0.2235</td>
<td>0.1608</td>
<td>0.2588</td>
<td>0.2588</td>
<td>0.1608</td>
</tr>
<tr>
<td></td>
<td>0.1608</td>
<td>0.2588</td>
<td>0.2588</td>
<td>0.2588</td>
<td>0.2588</td>
</tr>
</tbody>
</table>
CONCEPTS

- Bwlabel
- Bwselect
- Imerode
- Imdilate

- Imclose
- Imopen
- Regionprops
- Imrotate

These functions only can work with binary images, however Imrotate is the only function we can use with every kinds of images.
PROCEDURE
PROCEDURE

- Part 1
- Part 2
- Part 3
- Calculate limit
- How rotate
- Rotate 1 (or 2)
- Quit borders
- Margin
- Marco
- Main program
Part2
Part3
Calculate limit

*Calculate_limit_left.m*

Caculates im3, which will be important.
How rotate
How rotate

How_rotate_left.m
if sense_left==['SAR']
    if sense_right==['SAR']
        turned=rotate_1(A,im3_left,im3_right);
    else sense_right~==['SAR']
        turned=A;
    end
else sense_left==['NOSAR']
    if sense_right==['NOSAR']
        turned=rotate_2(A,im3_left,im3_right);
    else sense_right~==['NOSAR']
        turned=A;
    end
end
Rotate
Quit borders
Quit borders
Margins
Marco
NOISE IN IMAGES
Noise in Images

- Noise in images can be due to many causes (electronic components, external sources...)
- Depending on this causes we can use different methods
Noise in Images

- Linear filtering
- Median filtering
- Adaptative filtering
Linear filtering

- Linear filtering is filtering in which the value of an output pixel is a linear combination of the values of the pixels in the input pixel's neighborhood.

- $P_{xy} = a_1 * P_{x-1, y} + a_2 * P_{x-2, y} + ... + a_n * P_{x-n, y}$
Median filtering

- Average and median filters
- Difference between median and average

<table>
<thead>
<tr>
<th>1/9</th>
<th>1/9</th>
<th>1/9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/9</td>
<td>1/9</td>
<td>1/9</td>
</tr>
<tr>
<td>1/9</td>
<td>1/9</td>
<td>1/9</td>
</tr>
</tbody>
</table>
Wiener filter (a type of linear filter)

- Large variance => little smoothing
- Small variance => big smoothing

Wiener filter works better than linear filtering but Wiener filter requires more computation time.
Conclusions noise

- Depending on the kind of noise, we’ll apply different techniques of noise removal.

- Different kinds of noises are created by different causes: electronic components, natural sources, human sources...
PROBABILITY AND REDUNDANCY IN THE INFORMATION OF THE IMAGES
Entropy

- Quantity of information in a message
- How we can compress the image

\[ H(X) = E(I(X)) = \sum_{i=1}^{n} p(x_i) \log_2 \left( \frac{1}{p(x_i)} \right) \]

- Entropy is in bits (number of bits we’re going to codify the image)
Entropy in our images
Entropy in our images

![Graph showing entropy values for 3 components over images](image-url)
Histograms of one image
Conclusions Redundancy

- Blue component are more random that the rest.
- We would need more bits to compress the blue component.
- These differences bewteen blue component and the other are small.
GENERAL CONCLUSIONS
General conclusions

- The variety of the images has been the main problem.
- People work less, computers work harder.
- Quality better than comfort.
- Imwrite (image, 'path', 'quality', x).
ありがとうございます